THE UNIVERSITY of EDINBURGH
informatics

Applied Machine Learning (AML)

Non-Linear Dimensionality Reduction

Oisin Mac Aodha • Siddharth N.

Outline

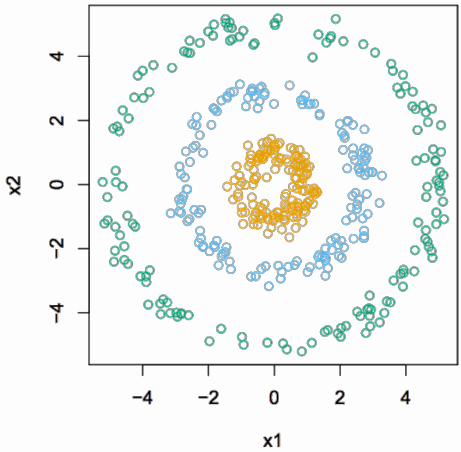
Non-Linearity

- Extensions for Linear Dimensionality Reduction
  - Kernel PCA
- Visualisation
  - Multi-Dimensional Scaling (MDS)
  - Isomap
  - Locally Linear Embeddings (LLE)
  - t-distributed Stochastic Neighbour Embedding (t-SNE)
  - Uniform Manifold Approximation & Projection (UMAP)

Extensions for Linear Dimensionality Reduction

PCA on Non-Linear Data

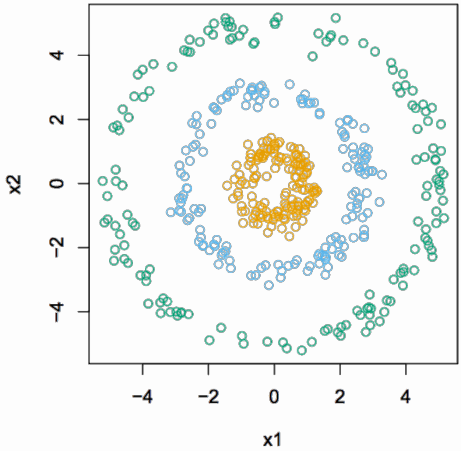
Example: Shells



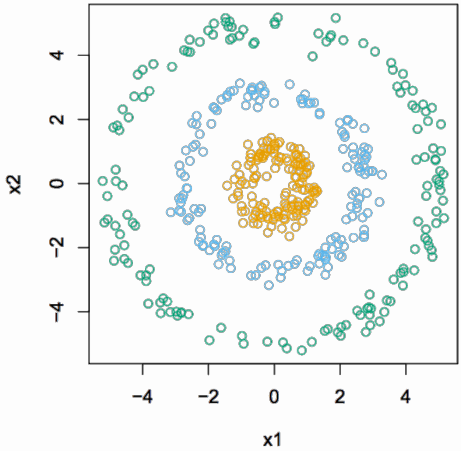
Data

PCA on Non-Linear Data

Example: Shells



Data



PCA

Feature Transformation

Key Idea: Transform inputs $\boldsymbol{x}$ using a feature map $\boldsymbol{\phi}(\boldsymbol{x})$

Feature Transformation

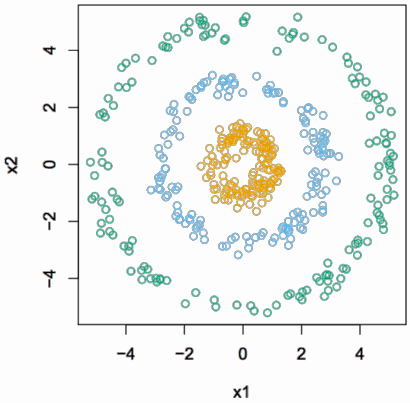
Key Idea: Transform inputs $\boldsymbol{x}$ using a feature map $\phi(\boldsymbol{x})$

$$\boldsymbol{x} \in \mathbb{R}^D,\ \phi(\boldsymbol{x}) \in \mathbb{R}^C,\ C > D!$$

Feature Transformation

Key Idea: Transform inputs $\boldsymbol{x}$ using a feature map $\phi(\boldsymbol{x})$

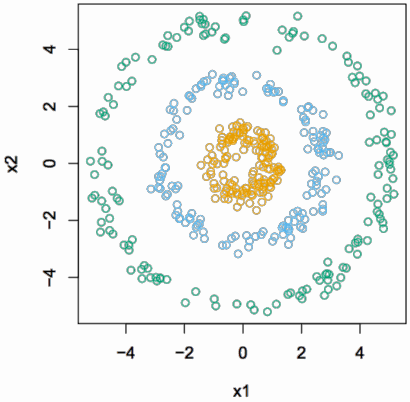
$$\boldsymbol{x} \in \mathbb{R}^D,\ \phi(\boldsymbol{x}) \in \mathbb{R}^C,\ C > D!$$

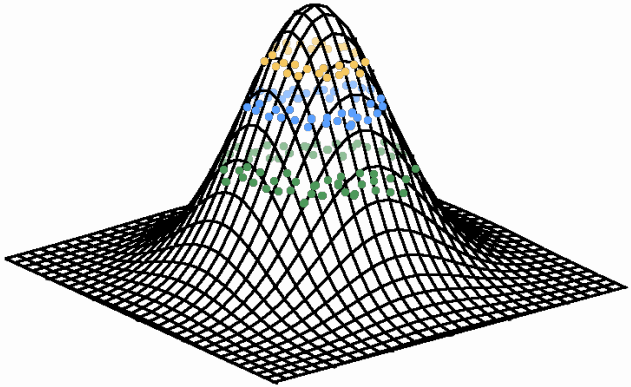


Feature Transformation

Key Idea: Transform inputs $\boldsymbol{x}$ using a feature map $\phi(\boldsymbol{x})$

$$\boldsymbol{x} \in \mathbb{R}^D,\ \phi(\boldsymbol{x}) \in \mathbb{R}^C,\ C > D!$$





Kernel PCA

$$X = [\boldsymbol{x}_1; \ldots; \boldsymbol{x}_N], \qquad \Phi(X) = [\phi(\boldsymbol{x}_1); \ldots; \phi(\boldsymbol{x}_N)]$$

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j) \qquad K = \Phi(X)^\top \Phi(X)$$

(kernel function) (kernel matrix)

Kernel PCA

$$X = [\boldsymbol{x}_1; \ldots; \boldsymbol{x}_N], \qquad \Phi(X) = [\phi(\boldsymbol{x}_1); \ldots; \phi(\boldsymbol{x}_N)]$$

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j) \qquad K = \Phi(X)^\top \Phi(X)$$

(kernel function) (kernel matrix)

$$S = \frac{1}{N} X X^\top$$

$$S\boldsymbol{v} = \lambda \boldsymbol{v}$$

Kernel PCA

$$X = [\boldsymbol{x}_1; \ldots; \boldsymbol{x}_N],$$

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)$$

(kernel function)

$$\Phi(X) = [\phi(\boldsymbol{x}_1); \ldots; \phi(\boldsymbol{x}_N)]$$

$$K = \Phi(X)^\top \Phi(X)$$

(kernel matrix)

$$S = \frac{1}{N} X X^\top$$

$$S\boldsymbol{v} = \lambda \boldsymbol{v}$$

$$S = \frac{1}{N} \Phi(X) \Phi(X)^\top$$

$$\boldsymbol{v} = \Phi(X)\boldsymbol{a} = \sum_{i=1}^{N} a_i \phi(\boldsymbol{x}_i)$$

(…some algebra)

$$K\boldsymbol{a} = N\lambda \boldsymbol{a}$$

Kernel PCA

$$X = [\boldsymbol{x}_1; \ldots; \boldsymbol{x}_N],$$

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)$$

(kernel function)

$$\Phi(X) = [\phi(\boldsymbol{x}_1); \ldots; \phi(\boldsymbol{x}_N)]$$

$$K = \Phi(X)^\top \Phi(X)$$

(kernel matrix)

$$S = \frac{1}{N} X X^\top$$

$$S\boldsymbol{v} = \lambda \boldsymbol{v}$$

$$S = \frac{1}{N} \Phi(X) \Phi(X)^\top$$

$$\boldsymbol{v} = \Phi(X)\boldsymbol{a} = \sum_{i=1}^{N} a_i \phi(\boldsymbol{x}_i)$$

(…some algebra)

$$K\boldsymbol{a} = N\lambda \boldsymbol{a}$$

Kernel Trick

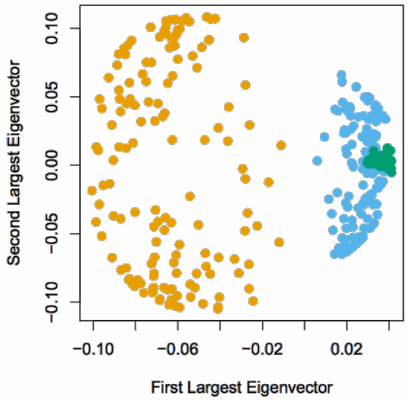
No need to compute $\phi(\boldsymbol{x})$—only $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$!

Kernel PCA: Example

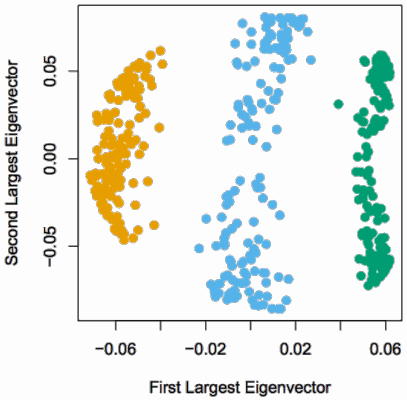
$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right) \qquad \text{(RBF Kernel)}$$

Kernel PCA: Example

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$$ (RBF Kernel)



RBF ($\sigma^2 = 0.5$)



RBF ($\sigma^2 = 0.01$)

Kernel PCA

Choosing the right kernel

- Not an easy choice

Kernel PCA

Choosing the right kernel

- Not an easy choice
- Construct by hand/eye where feasible

Kernel PCA

Choosing the right kernel

- Not an easy choice
- Construct by hand/eye where feasible
- Possible to learn kernel matrices K directly from data!

Kernel PCA

Choosing the right kernel

- Not an easy choice
- Construct by hand/eye where feasible
- Possible to learn kernel matrices K directly from data!

Kernel PCA

Choosing the right kernel

- Not an easy choice
- Construct by hand/eye where feasible
- Possible to learn kernel matrices K directly from data!

Several non-linear dimensionality reduction methods can be viewed as kernel PCA, with kernels learned from data [1]

1. J. Ham et al, A Kernel View of the Dimensionality Reduction of Manifolds, 2004

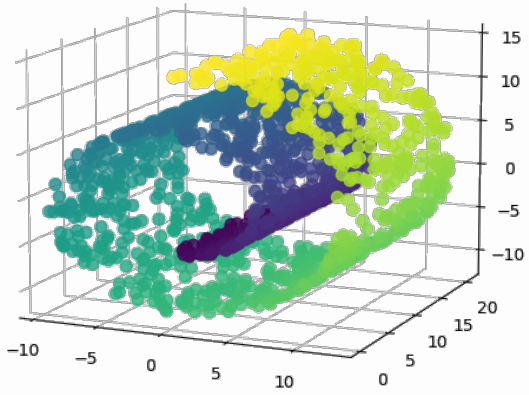
Visualisation

Manifold Hypothesis

High-dimensional data in the real world really lies on low-dimensional manifolds within that high-dimensional space.

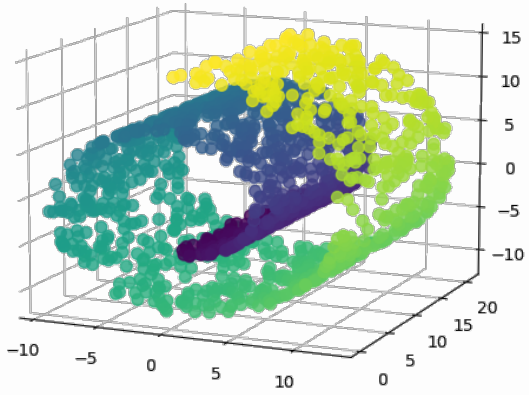
Manifold Hypothesis

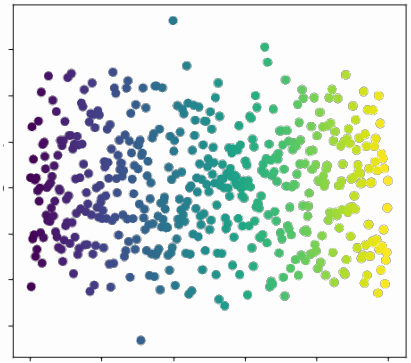
High-dimensional data in the real world really lies on low-dimensional manifolds within that high-dimensional space.



Manifold Hypothesis

High-dimensional data in the real world really lies on low-dimensional manifolds within that high-dimensional space.





Overview

Key Ideas

- Difficult to construct a single global transformation of data
- Focus instead on some *local* measure of closeness
- Project data x onto lower-dimensional manifold as e
- **Question:** can we *preserve* local measure of closeness?

Overview

Key Ideas

- Difficult to construct a single global transformation of data
- Focus instead on some *local* measure of closeness
- Project data $\boldsymbol{x}$ onto lower-dimensional manifold as $\boldsymbol{e}$
- **Question:** can we *preserve* local measure of closeness?

Let $\mathcal{X}$ denote the high-dimensional data space, and $\mathcal{E}$ denote the low-dimensional manifold space. We can define the following distance measures on the two spaces respectively

$$\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \qquad \mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_j)$$

Multi-Dimensional Scaling (MDS)

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the distance between *every pair of samples in original data* X

Multi-Dimensional Scaling (MDS)

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the distance between *every pair of samples in original data* X

$$\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \qquad \mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_j) = \|\boldsymbol{e}_i - \boldsymbol{e}_j\| \qquad \text{(example)}$$

Multi-Dimensional Scaling (MDS)

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the distance between *every pair of samples in original data* X

$$\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \qquad \mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_j) = \|\boldsymbol{e}_i - \boldsymbol{e}_j\| \qquad \text{(example)}$$

Objective: $\min \sum_{i,j} (\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_j))^2$

Multi-Dimensional Scaling (MDS)

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the distance between *every pair of samples in original data* X

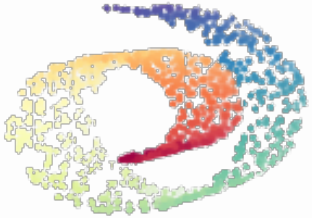
$$\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \qquad \mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_j) = \|\boldsymbol{e}_i - \boldsymbol{e}_j\| \qquad \text{(example)}$$

$$\text{Objective:} \quad \min \sum_{i,j} (\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_j))^2$$

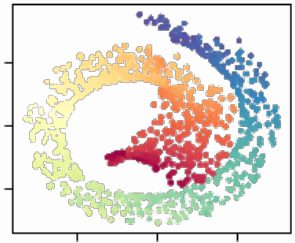
- distance metrics $\mathcal{D}_{\mathcal{X}}, \mathcal{D}_{\mathcal{E}}$ could really be anything
- choosing L_2 helps make optimisation simpler

MDS: Example

Swiss Roll



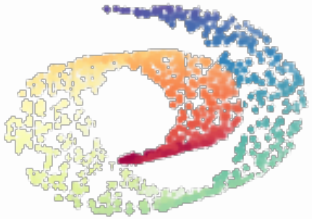
Data



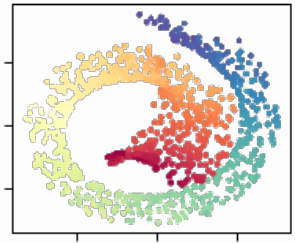
MDS (Euclidean)

MDS: Example

Swiss Roll



Data



MDS (Euclidean)

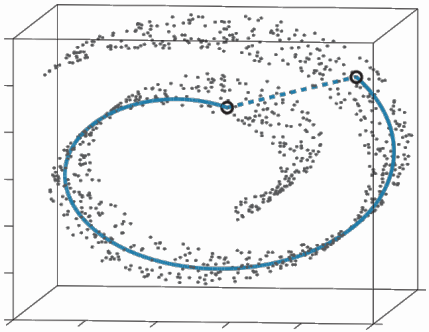
- Projects down to 2D while preserving distances
- Preserving distances *outside* the manifold!

Isomap

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the distance between points on the *embedded manifold*, not arbitrary distance!

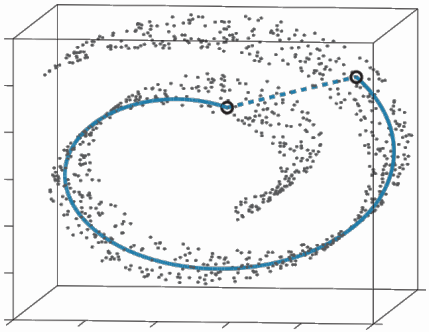
Isomap

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the distance between points on the *embedded manifold*, not arbitrary distance!



Isomap

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the distance between points on the *embedded manifold*, not arbitrary distance!

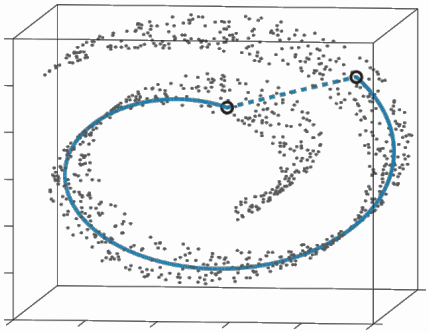


Find the Geodesic

- Generate nearest-neighbour graph $\mathcal{G}$ on data

Isomap

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the distance between points on the *embedded manifold*, not arbitrary distance!

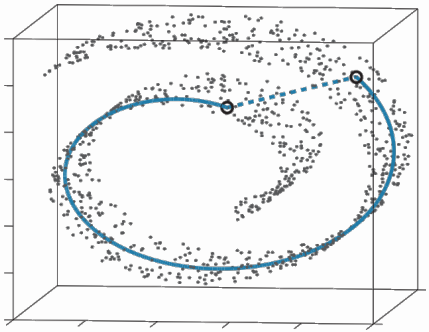


Find the Geodesic

- Generate nearest-neighbour graph $\mathcal{G}$ on data
- Shortest distance between points in this graph
  - Floyd-Warshall algorithm (*all pairs* shortest path)

Isomap

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the distance between points on the *embedded manifold*, not arbitrary distance!

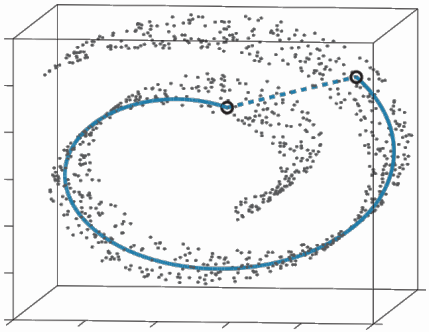


Find the Geodesic

- Generate nearest-neighbour graph $\mathcal{G}$ on data
- Shortest distance between points in this graph
  - Floyd-Warshall algorithm (*all pairs* shortest path)
- Perform MDS with $\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathcal{G}_{\text{FW}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Isomap

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the distance between points on the *embedded manifold*, not arbitrary distance!

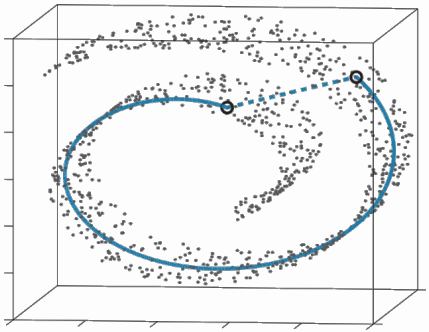


Find the Geodesic

- Generate nearest-neighbour graph $\mathcal{G}$ on data
- Shortest distance between points in this graph
  - Floyd-Warshall algorithm (*all pairs* shortest path)
- Perform MDS with $\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathcal{G}_{\text{FW}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Isomap

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the distance between points on the *embedded manifold*, not arbitrary distance!



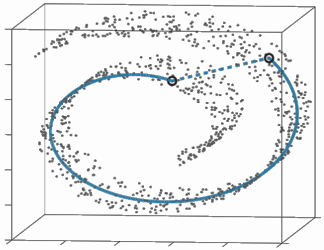
Find the Geodesic

- Generate nearest-neighbour graph $\mathcal{G}$ on data
- Shortest distance between points in this graph
  - Floyd-Warshall algorithm (*all pairs* shortest path)
- Perform MDS with $\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathcal{G}_{\text{FW}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$

Objective: $\min \sum_{i,j} (\mathcal{G}_{\text{FW}}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \|\boldsymbol{e}_i - \boldsymbol{e}_j\|)^2$

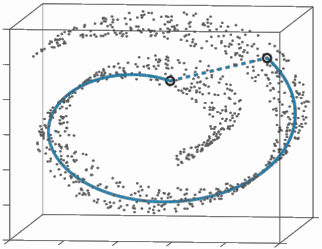
J.B. Tenenbaum et al, A Global Geometric Framework for Nonlinear Dimensionality Reduction, 2000

Isomap: Manifold

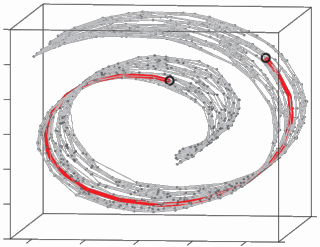


Data

Isomap: Manifold



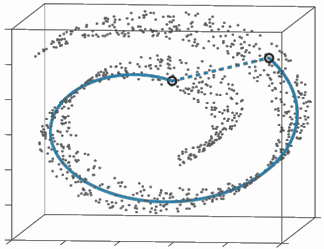
Data



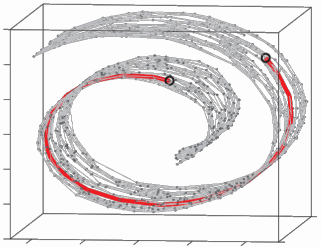
Graph

+ Floyd-Warshall

Isomap: Manifold

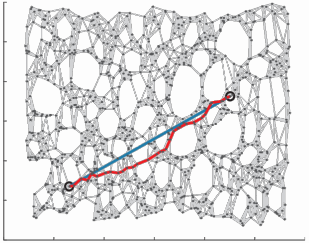


Data



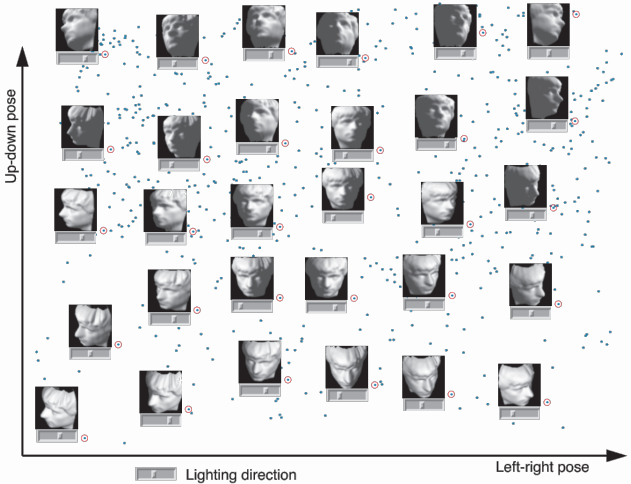
Graph

+ Floyd-Warshall

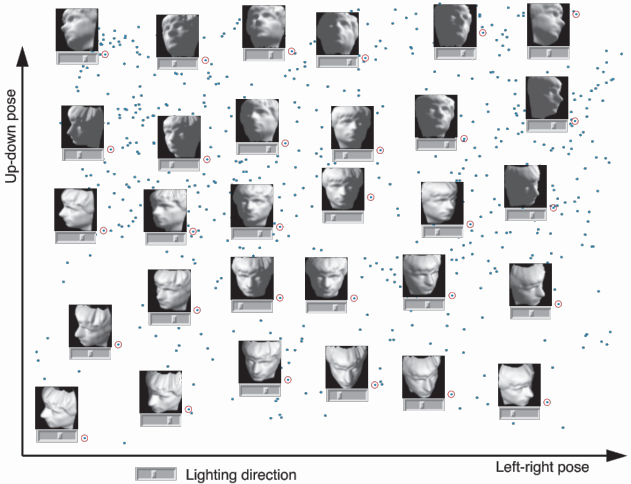


MDS

Isomap: Example and Issues



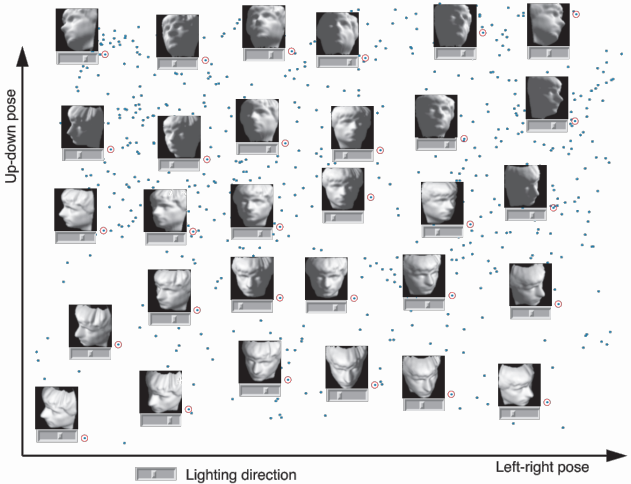
Isomap: Example and Issues



Issues

- requires *uniform* and *dense* sampling on manifold

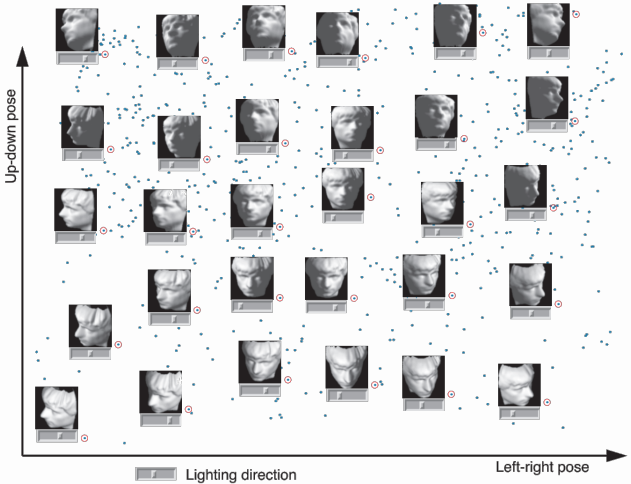
Isomap: Example and Issues



Issues

- requires *uniform* and *dense* sampling on manifold
- prone to topological instabilities (effect of noise, non-convexity)

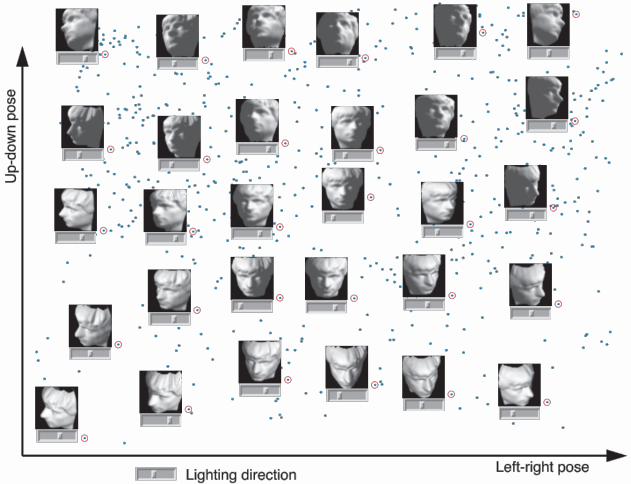
Isomap: Example and Issues



Issues

- requires *uniform* and *dense* sampling on manifold
- prone to topological instabilities (effect of noise, non-convexity)
- can get disconnected graphs!

Isomap: Example and Issues



Issues

- requires *uniform* and *dense* sampling on manifold
- prone to topological instabilities (effect of noise, non-convexity)
- can get disconnected graphs!
- slow with size of data — Floyd-Warshall is $O(N^3)$!

Locally Linear Embeddings (LLE)

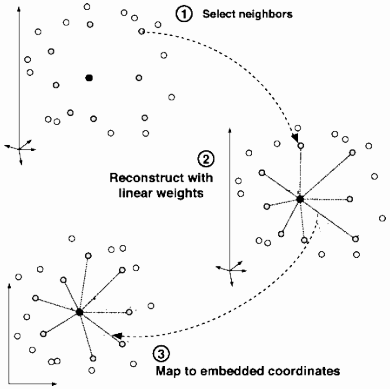
Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the *linear transform that reconstructs points* from the K nearest neighbours

Locally Linear Embeddings (LLE)

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the *linear transform that reconstructs points* from the K nearest neighbours

$$\arg\min_{w_1,\ldots,w_K} \sum_{i=1}^{N} (\boldsymbol{x}_i - \sum_{\boldsymbol{x}_k \in n(\boldsymbol{x}_i)} w_k \boldsymbol{x}_k)^2$$

$$\arg\min_{\boldsymbol{e}_1,\ldots,\boldsymbol{e}_N} \sum_{i=1}^{N} (\boldsymbol{e}_i - \sum_{\boldsymbol{e}_k \in n(\boldsymbol{e}_i)} w_k \boldsymbol{e}_k)^2$$



Algorithm

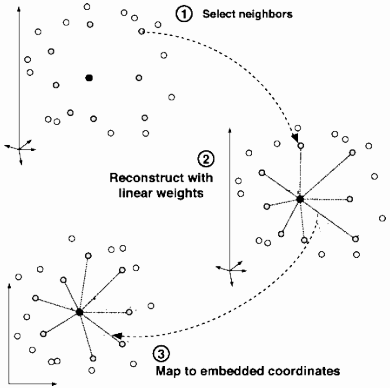
1. Find the weights w_k
2. Fix weights w_k and find optimal embeddings $\boldsymbol{e}_i$ solved using a (sparse) eigen-decomposition

Locally Linear Embeddings (LLE)

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the *linear transform that reconstructs points* from the K nearest neighbours

$$\arg\min_{w_1,\ldots,w_K} \sum_{i=1}^{N} (\boldsymbol{x}_i - \sum_{\boldsymbol{x}_k \in n(\boldsymbol{x}_i)} w_k \boldsymbol{x}_k)^2$$

$$\arg\min_{\boldsymbol{e}_1,\ldots,\boldsymbol{e}_N} \sum_{i=1}^{N} (\boldsymbol{e}_i - \sum_{\boldsymbol{e}_k \in n(\boldsymbol{e}_i)} w_k \boldsymbol{e}_k)^2$$

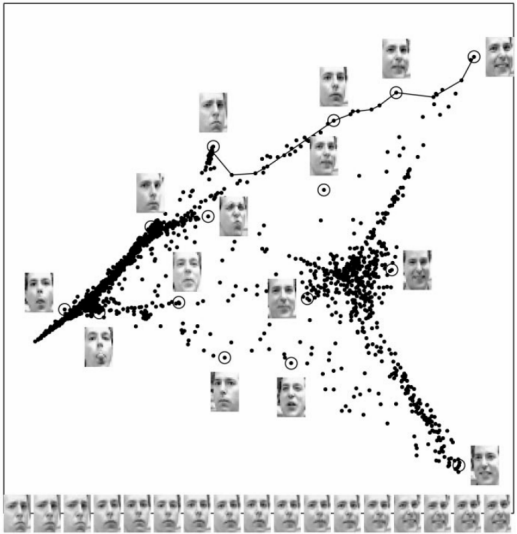


Algorithm

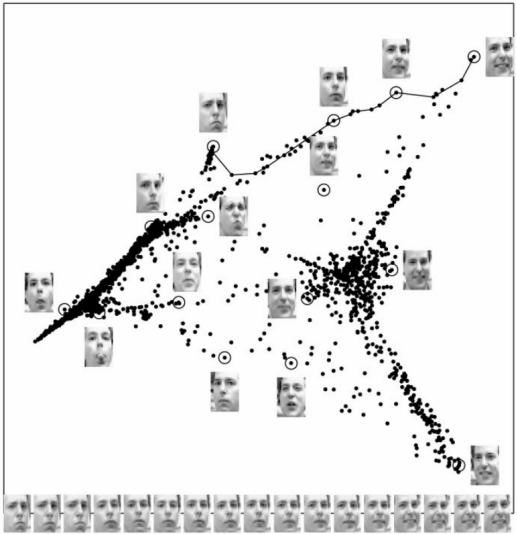
1. Find the weights w_k
2. Fix weights w_k and find optimal embeddings $\boldsymbol{e}_i$ solved using a (sparse) eigen-decomposition

S.T. Roweis & L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, 2000

LLE: Example and Issues



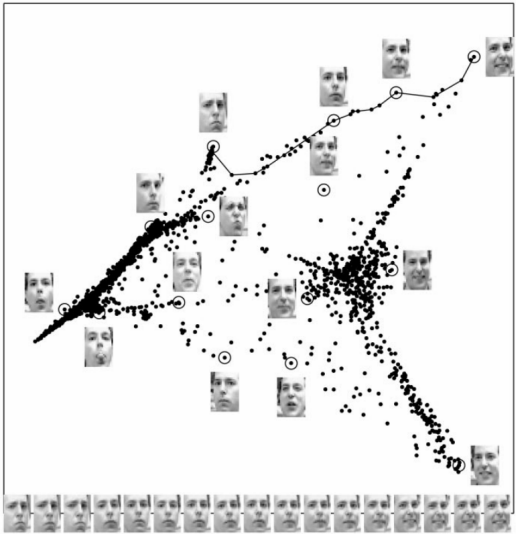
LLE: Example and Issues



Advantages

- globally optimal

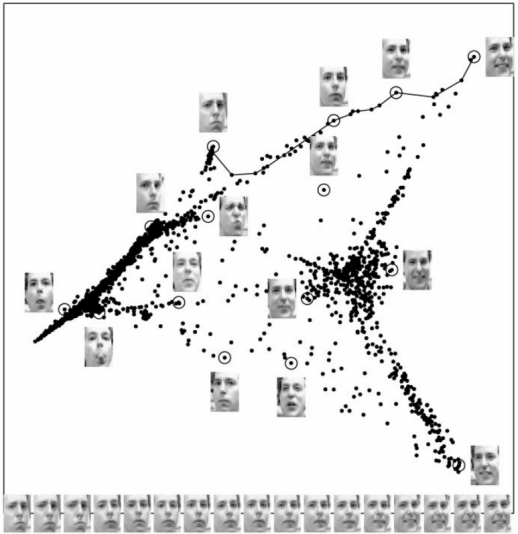
LLE: Example and Issues



Advantages

- globally optimal
- only bottleneck—finding $\boldsymbol{e}_i$

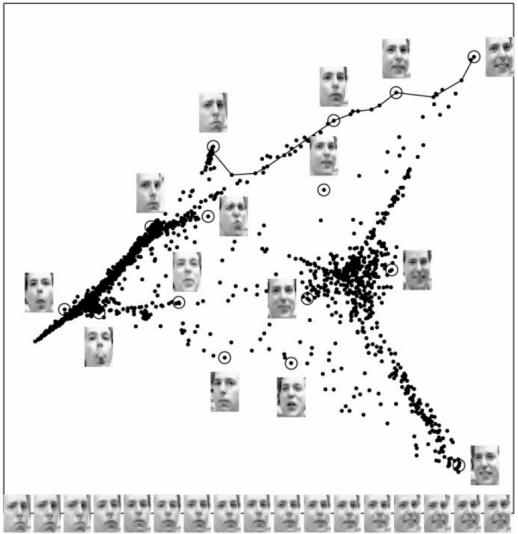
LLE: Example and Issues



Advantages

- globally optimal
- only bottleneck—finding $\boldsymbol{e}_i$
- *not* preserving specific distance

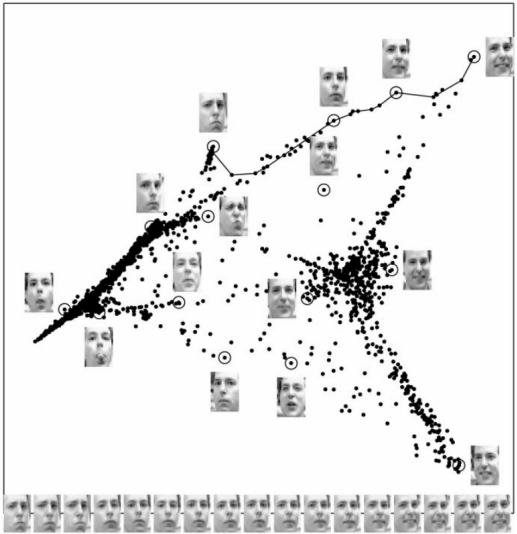
LLE: Example and Issues



Advantages

- globally optimal
- only bottleneck—finding $\boldsymbol{e}_i$
- *not* preserving specific distance

LLE: Example and Issues



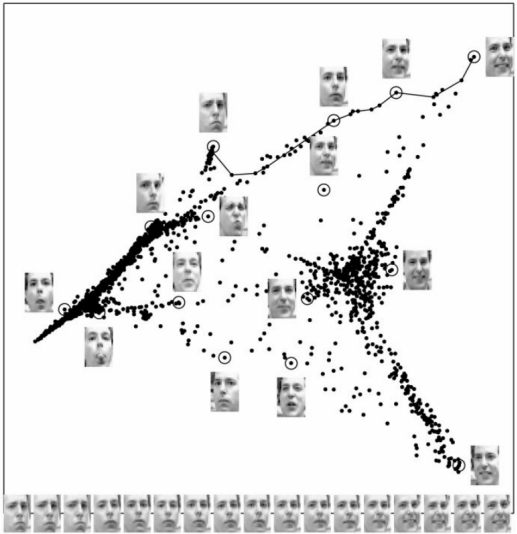
Advantages

- globally optimal
- only bottleneck—finding $\boldsymbol{e}_i$
- *not* preserving specific distance

Issues

- reliance on local smoothness

LLE: Example and Issues



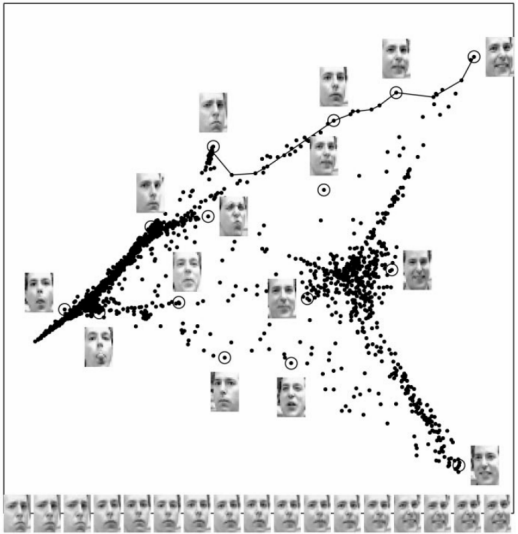
Advantages

- globally optimal
- only bottleneck—finding $\boldsymbol{e}_i$
- *not* preserving specific distance

Issues

- reliance on local smoothness
- sensitive to noise

LLE: Example and Issues



Advantages

- globally optimal
- only bottleneck—finding $\boldsymbol{e}_i$
- *not* preserving specific distance

Issues

- reliance on local smoothness
- sensitive to noise
- no theoretical guarantees about manifold

t-distributed Stochastic Neighbour Embedding (t-SNE)

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the *probability distribution over pairwise similarities* between points in the space.

t-distributed Stochastic Neighbour Embedding (t-SNE)

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the *probability distribution over pairwise similarities* between points in the space.

$$p(\boldsymbol{x}_j|\boldsymbol{x}_i) = \frac{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2/2\sigma_i^2)} \quad (i \neq j)$$

$$p(\boldsymbol{x}_i|\boldsymbol{x}_i) = 0 \qquad \sum_i p(\boldsymbol{x}_j|\boldsymbol{x}_i) = 1$$

$$\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{2N}\left(p(\boldsymbol{x}_i|\boldsymbol{x}_j) + p(\boldsymbol{x}_j|\boldsymbol{x}_i)\right)$$

$$\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_i) = 0 \qquad \sum_{ij} \mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1$$

t-distributed Stochastic Neighbour Embedding (t-SNE)

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the *probability distribution over pairwise similarities* between points in the space.

$$p(\boldsymbol{x}_j|\boldsymbol{x}_i) = \frac{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2/2\sigma_i^2)} \quad (i \neq j)$$

$$p(\boldsymbol{x}_i|\boldsymbol{x}_i) = 0 \qquad \sum_i p(\boldsymbol{x}_j|\boldsymbol{x}_i) = 1$$

$$\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{2N}\left(p(\boldsymbol{x}_i|\boldsymbol{x}_j) + p(\boldsymbol{x}_j|\boldsymbol{x}_i)\right)$$

$$\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_i) = 0 \qquad \sum_{ij} \mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1$$

$$\mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_j) := \text{Student}\text{-}t(\|\boldsymbol{e}_i - \boldsymbol{e}_j\|^2, \nu = 1)$$

$$\mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_j) = \frac{\left(1 + \|\boldsymbol{e}_i - \boldsymbol{e}_j\|^2\right)^{-1}}{\sum_k \sum_{l\neq k}\left(1 + \|\boldsymbol{e}_k - \boldsymbol{e}_l\|^2\right)^{-1}}$$

$$\mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_i) = 0$$

Student-t allows dissimilar points in $\mathcal{X}$ to be modelled far away in $\mathcal{E}$!

t-distributed Stochastic Neighbour Embedding (t-SNE)

Project data from $\mathcal{X}$ to $\mathcal{E}$ while preserving the *probability distribution over pairwise similarities* between points in the space.

$$p(\boldsymbol{x}_j|\boldsymbol{x}_i) = \frac{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2/2\sigma_i^2)} \quad (i \neq j)$$

$$p(\boldsymbol{x}_i|\boldsymbol{x}_i) = 0 \qquad \sum_i p(\boldsymbol{x}_j|\boldsymbol{x}_i) = 1$$

$$\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{2N}\left(p(\boldsymbol{x}_i|\boldsymbol{x}_j) + p(\boldsymbol{x}_j|\boldsymbol{x}_i)\right)$$

$$\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_i) = 0 \qquad \sum_{ij} \mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1$$

$$\mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_j) := \text{Student}\text{-}t(\|\boldsymbol{e}_i - \boldsymbol{e}_j\|^2, \nu = 1)$$

$$\mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_j) = \frac{\left(1 + \|\boldsymbol{e}_i - \boldsymbol{e}_j\|^2\right)^{-1}}{\sum_k \sum_{l\neq k}\left(1 + \|\boldsymbol{e}_k - \boldsymbol{e}_l\|^2\right)^{-1}}$$

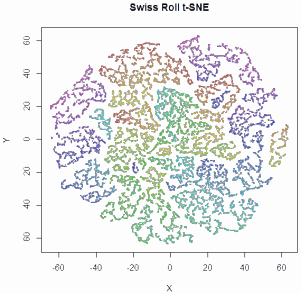
$$\mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_i) = 0$$

Student-t allows dissimilar points in $\mathcal{X}$ to be modelled far away in $\mathcal{E}$!

Objective: $\min \mathrm{KL}\big(\mathcal{D}_{\mathcal{X}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \| \, \mathcal{D}_{\mathcal{E}}(\boldsymbol{e}_i, \boldsymbol{e}_j)\big)$

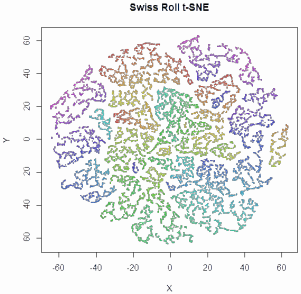
L.J.P. van der Maaten & G.E. Hinton, Visualizing Data using t-SNE, 2008

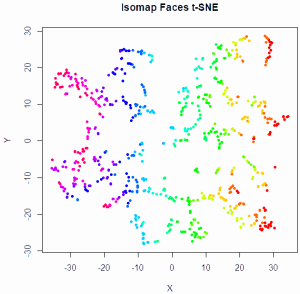
t-SNE: Examples



Figures: https://jlmelville.github.io/uwot/umap-examples.html

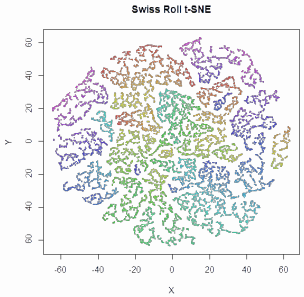
t-SNE: Examples

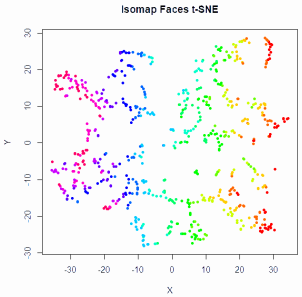


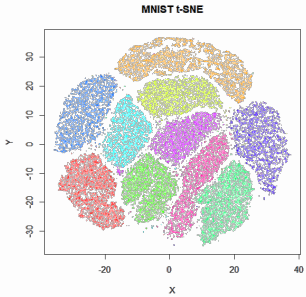


Figures: https://jlmelville.github.io/uwot/umap-examples.html

t-SNE: Examples







Figures: https://jlmelville.github.io/uwot/umap-examples.html

So Far

- Different (relatively simple) aspects may be preserved in mapping $\mathcal{X}$ to $\mathcal{E}$

So Far

- Different (relatively simple) aspects may be preserved in mapping $\mathcal{X}$ to $\mathcal{E}$
- Trade-off in terms of simplicity of assumption and ease of optimisation

So Far

- Different (relatively simple) aspects may be preserved in mapping $\mathcal{X}$ to $\mathcal{E}$
- Trade-off in terms of simplicity of assumption and ease of optimisation
- t-SNE quite popular
  - can incorporate both global and local structure (distributional)
  - works on large scale *and* high-dimensional data

So Far

- Different (relatively simple) aspects may be preserved in mapping $\mathcal{X}$ to $\mathcal{E}$
- Trade-off in terms of simplicity of assumption and ease of optimisation
- t-SNE quite popular
  - can incorporate both global and local structure (distributional)
  - works on large scale *and* high-dimensional data

So Far

- Different (relatively simple) aspects may be preserved in mapping $\mathcal{X}$ to $\mathcal{E}$
- Trade-off in terms of simplicity of assumption and ease of optimisation
- t-SNE quite popular
  - can incorporate both global and local structure (distributional)
  - works on large scale *and* high-dimensional data

Issue

None of these methods learn an *explicit* metric

- $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \xrightarrow{f} \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N\}$ but $\boldsymbol{x}_t \xrightarrow{\not{f}} \boldsymbol{e}_t$ for unseen $\boldsymbol{x}_t$
- cannot project an unseen point without redoing the optimisation!

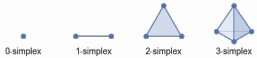
Uniform Manifold Approximation & Projection (UMAP)

Leverage Riemannian geometry and topology to construct a general framework for manifold learning and dimensionality reduction.

Uniform Manifold Approximation & Projection (UMAP)

Leverage Riemannian geometry and topology to construct a general framework for manifold learning and dimensionality reduction.

Overview



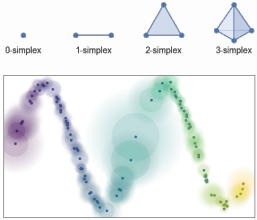
- Use simplices as basic building block

Uniform Manifold Approximation & Projection (UMAP)

Leverage Riemannian geometry and topology to construct a general framework for manifold learning and dimensionality reduction.

Overview

- Use simplices as basic building block
- Estimate connectivity and distances between points



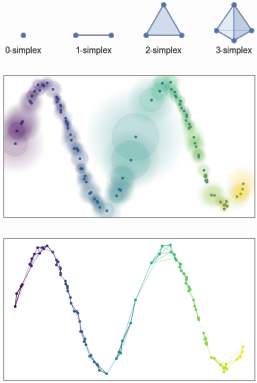
L. McInnes et al, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, 2018

Uniform Manifold Approximation & Projection (UMAP)

Leverage Riemannian geometry and topology to construct a general framework for manifold learning and dimensionality reduction.

Overview

- Use simplices as basic building block
- Estimate connectivity and distances between points
- Construct graph that captures topology of manifold!



L. McInnes et al, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, 2018

UMAP:Dimensionality Reduction

- Construct faithful topological representation of data

UMAP:Dimensionality Reduction

- Construct faithful topological representation of data
- Compute *cross-entropy* between topological structures of $\mathcal{X}$ and $\mathcal{E}$ in terms of the simplices (building blocks)

UMAP:Dimensionality Reduction

- Construct faithful topological representation of data
- Compute *cross-entropy* between topological structures of $\mathcal{X}$ and $\mathcal{E}$ in terms of the simplices (building blocks)
- Optimise low-dimensional representation to have minimum cross-entropy

UMAP:Dimensionality Reduction

- Construct faithful topological representation of data
- Compute *cross-entropy* between topological structures of $\mathcal{X}$ and $\mathcal{E}$ in terms of the simplices (building blocks)
- Optimise low-dimensional representation to have minimum cross-entropy

UMAP:Dimensionality Reduction

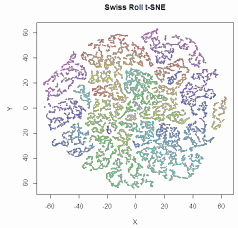
- Construct faithful topological representation of data
- Compute *cross-entropy* between topological structures of $\mathcal{X}$ and $\mathcal{E}$ in terms of the simplices (building blocks)
- Optimise low-dimensional representation to have minimum cross-entropy

Advantages

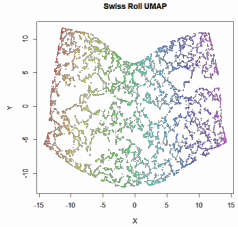
- Can learn a metric, so computing $\boldsymbol{x}_t \xrightarrow{f} \boldsymbol{e}_t$ for unseen $\boldsymbol{x}_t$ is feasible (when labels available)!
- Is fast and scalable
- Can be run unsupervised, supervised, or even weakly supervised!

UMAP: Examples and Comparisons

t-SNE

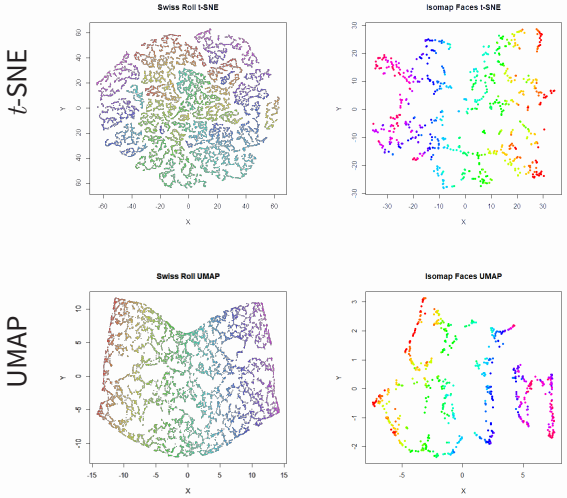


UMAP



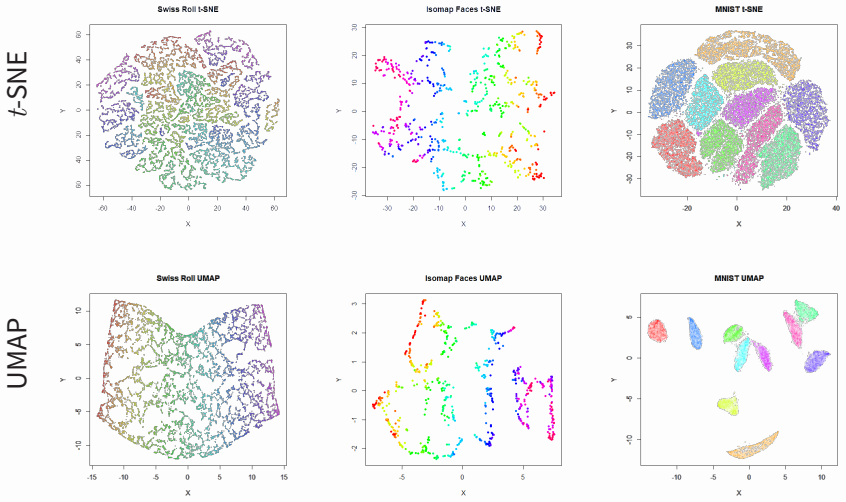
Figures: https://jlmelville.github.io/uwot/umap-examples.html

UMAP: Examples and Comparisons



Figures: https://jlmelville.github.io/uwot/umap-examples.html

UMAP: Examples and Comparisons



Figures: https://jlmelville.github.io/uwot/umap-examples.html

Summary

- Non-linear dimensionality reduction helps visualise complex data in low dimensions—relies on the manifold hypothesis

Summary

- Non-linear dimensionality reduction helps visualise complex data in low dimensions—relies on the manifold hypothesis
- Can explore ways to transform data to run linear versions of algorithms (e.g. kernel PCA)

Summary

- Non-linear dimensionality reduction helps visualise complex data in low dimensions—relies on the manifold hypothesis
- Can explore ways to transform data to run linear versions of algorithms (e.g. kernel PCA)
- Most methods exploit the nearest neighbour graph in some form (e.g. MDS, Isomap, LLE, etc.)

Summary

- Non-linear dimensionality reduction helps visualise complex data in low dimensions—relies on the manifold hypothesis
- Can explore ways to transform data to run linear versions of algorithms (e.g. kernel PCA)
- Most methods exploit the nearest neighbour graph in some form (e.g. MDS, Isomap, LLE, etc.)
- Data is typically required to be clean (not much noise) and dense …not too strong a requirement, especially with vision or language

Summary

- Non-linear dimensionality reduction helps visualise complex data in low dimensions—relies on the manifold hypothesis
- Can explore ways to transform data to run linear versions of algorithms (e.g. kernel PCA)
- Most methods exploit the nearest neighbour graph in some form (e.g. MDS, Isomap, LLE, etc.)
- Data is typically required to be clean (not much noise) and dense …not too strong a requirement, especially with vision or language
- Many approaches to choose from—t-SNE and UMAP most popular